REMARKS

Claims 1-10 are pending in this application. Claims 1-10 are independent. In light of the remarks made herein, Applicants respectfully request reconsideration and withdrawal of the outstanding rejections.

In the outstanding Official Action, the Examiner rejected claims 1-5 under 35 U.S.C. § 102(e) as being anticipated by *Kohara* (USP 6,600,575); rejected claims 6-7 under 35 U.S.C. § 102(e) as being anticipated by *Iwazaki* (USP 6,073,244); rejected claims 8 and 10 under 35 U.S.C. § 103(a) as being unpatentable over *Iwazaki* in view of *Dowling* (USP 6,226,738); and rejected claim 9 under 35 U.S.C. § 103(a) as being unpatentable over *Iwazaki* in view of *Kapusta et al.* (USP 5,799,176). Applicants respectfully traverse these rejections.

Claim Rejections - 35 U.S.C. § 102(e) - *Kohara*

The Examiner rejected claims 1-5, asserting *Kohara* anticipates the invention as set forth in these claims. Specifically, the Examiner asserts that *Kohara* teaches, for example, with regard to claim 1, a clock supply circuit, citing to col. 1, lines 22-33, and a clock selector circuit, citing to col. 2, lines 13-33 and col. 4, lines 37-65. Applicants respectfully disagree with the Examiner's characterization of this reference.

The disclosure of *Kohara* is directed to a clock supply circuit. The clock supply circuit includes a clock dividing section that receives a system clock and generates and outputs clock signals of two or more types. Selectors select any of the clock signals of the two or more types outputted by the clock dividing section and feed it to a printing control block or reading control block (Abstract). Specifically, as set forth in col. 5, line 63 - col. 6, line 5, *Kohara* teaches as follows:

> According to the clock supply circuit of the present invention, a clock frequency to be supplied to each of the control blocks 2, 3 and 4 within the custom IC 5 is changed depending on each operation corresponding to each operation mode of the facsimile. This allows unnecessary power consumption in each block in a non-operation state at idle to be reduced and a noise from each block to be controlled. To achieve this, it is necessary to decide in what kinds of operational states the facsimile is. The method for the decision is described below.

Further, as disclosed in Figs. 5-7 and 9, and col. 6, lines 33-64, *Kohara* suggests the clock be changed at the start of reading the first line of the reading page and at the end of reading the final line of the reading page.

In contrast, the present invention as set forth in claim 1 recites, *inter alia*, a semiconductor integrated circuit having one or more functional circuit blocks and executing a set of instructions, comprising a clock selector circuit that selects a clock being fed to each of the functional circuit blocks for each execution cycle, wherein the clock supply circuit and the clock selector circuit are configured so as to change an operating frequency or halt operation of the one or more functional circuit blocks for each execution cycle.

As noted above, *Kohara* teaches supplying a clock frequency depending upon each operation corresponding to each operation mode of the facsimile. As explained in col. 6, lines 15-29, one clock may be supplied to the reading control block while another clock may be supplied to the printing control block. However, there is no teaching or suggestion in *Kohara* that is directed to selecting a clock being fed to each of the functional circuit blocks **for each execution cycle**. As *Kohara* fails to teach or suggest all of the elements as set forth in claim 1, *Kohara* fails to anticipate the present invention.

It appears that the Examiner is construing the meaning of the term "execution cycle" in a way that is inconsistent with the meaning of this term as set forth in the specification and as is known to one of ordinary skill in the art. As can be seen in the specification, for example, in paragraphs [00054] and [00055], Applicants provide a definition for the term "execution cycle." Additionally, as can be seen from the attached document in Appendix A entitled "Instruction Execution Cycle," it is well known that an execution cycle can be broken down into a "fetch cycle" and an "execute cycle." In considering the appropriate definition of the term "execution cycle," it is respectfully submitted that *Kohara* fails to teach the clock selector circuit of the present invention that selects a clock being fed to each of the functional circuit blocks **for each execution cycle**, wherein the clock supply circuit and the clock selector circuit are configured so as to **change an operating frequency or halt operation** of the one or more functional circuit blocks **for each execution cycle** as set forth in claim 1. As such, *Kohara* fails to anticipate claim 1 as *Kohara* fails to teach or suggest all of the claim elements.

It is respectfully submitted that claims 2-5 contain elements similar to those discussed above with regard to claim 1, and thus claims 2-5 are not anticipated by *Kohara* for the reasons set forth above with regard to claim 1.

With regard to the Examiner's rejection of claim 2, the Examiner asserts *Kohara* teaches an instruction decoded circuit that feeds a selection signal to the clock selector circuit for selecting a most appropriate clock from one or more clocks by analyzing prescribed bits of an instruction code, citing to col. 10, line 56 through col. 11, line 13. However, Applicants respectfully submit that there is no teaching or suggestion in *Kohara* that is directed to this claim element. As such, it is respectfully requested that, as the cited reference fails to teach or suggest all of the claim elements, the outstanding rejection be withdrawn.

With regard to the Examiner's rejection of claim 3, the Examiner asserts *Kohara* teaches the memory select signal circuit citing to the reading function and Fig. 1. Applicants respectfully disagree with the Examiner's assertions.

Kohara teaches the clock C2 is to be fed to a reading control block 2 (Figs. 1 and 2). *Kohara* does not disclose that the clock frequency to be supplied to the memory itself is changed. As such, there is no teaching or suggestion in *Kohara* that is directed to a memory select signal circuit that identifies a memory block to be accessed, wherein the clock supply circuit and the clock selector circuit are configured so as to change an operating frequency or halt operation of the semiconductor integrated circuit, a part thereof, said one or more memory blocks, or the single memory block for each execution cycle in accordance with performance of the memory block that is identified by the memory select signal circuit as recited in claim 3. As such, it is respectfully submitted that *Kohara* fails to anticipate claim 3.

With regard to the Examiner's rejection of claim 4, the Examiner asserts *Kohara* teaches an I/O select circuit that identifies a peripheral circuit to be accessed citing to the printing function and Fig. 1. Applicants respectfully disagree with the Examiner's assertions.

It is respectfully submitted that *Kohara* teaches clock C3 fed to the printing control block 3 is for control operations by the printing control block 3. *Kohara* does not disclose an I/O select signal circuit that identifies a peripheral circuit to be accessed, wherein the clock supply circuit and the clock selector circuit are configured so as to change an operating frequency or halt operation of the semiconductor integrated circuit, a part thereof, or said one or more peripheral circuits for each execution cycle in accordance with performance of the peripheral circuit that is identified by the I/O select signal circuit. As such, it is respectfully requested that the outstanding rejection be withdrawn.

Claim Rejections - 35 U.S.C. § 102(e) - *Iwazaki*

With regard to the Examiner's rejection of claim 6, the Examiner asserts that *Iwazaki* teaches all of the elements as set forth in the claims, citing to col. 3, line 48 - col. 4, line 61. Applicants respectfully disagree with the Examiner's characterization of this reference.

The disclosure of *Iwazaki* is directed to a power saving clock control apparatus and method. A clock control type information processing apparatus of the invention selects clock frequency according to a load state which reduces electric power consumption without substantially reducing the effect of performance of the program. The clock control type information processing apparatus includes a CPU for executing programs and a plurality of peripheral processing units connected to the central processing unit using a bus, includes a clock generating unit for generating clock signals having a plurality of frequencies and selectively supplying any one of the clock signals to the central processing unit and the peripheral processing unit, a bus access monitoring unit for monitoring load state of the bus which connects the central processing unit with the peripheral processing units, and a clock selection control unit for generating control signals to control the clock frequencies generated by the clock generating unit according to the load state of the bus (Abstract).

Further, *Iwazaki* teaches a "predetermined period", i.e., monitoring the bus access of the peripheral processing unit according to the load thus monitored and turning on/or the clocks supplied to the peripheral processing unit, if necessary. This "predetermined period" is a substantially long time when compared to the execution cycle of the present invention. Further, *Iwazaki* teaches that the clocks are controlled for the peripheral processing unit and the CPU.

In contrast, the present invention as set forth in claim 6 recites, *inter alia*, a semiconductor integrated circuit having one or more functional circuit blocks and executing at least either of data processing or instruction processing in a pipeline having a plurality of stages when running a set of instructions, comprising a clock supply circuit that supplies one or more clocks for driving the stages at a different frequency; a clock selector circuit that selects a different clock to be given to each of the stages for each execution cycle; and an analyzer circuit wherein the analyzer circuit is arranged so as to analyze the instructions to be executed in each stage of the pipeline and feeds a signal to the clock selector circuit so that when the stages execute the instructions having a load different from each other, a stage executing an instruction having a lighter load is provided with a slower clock.

As noted above, *Iwazaki* discloses selecting a particular clock depending upon the load state of the bus which connects the central processing unit with the peripheral processing units, and that the clocks are controlled for the peripheral processing unit and the CPU. There is no discussion in *Iwazaki* that is directed to a pipeline having a plurality of stages. Further, there is no discussion in *Iwazaki* that is directed to a clock selector circuit that selects a different clock to be given to each of the stages for each execution cycle of the pipeline. Finally, there is no teaching or suggestion in *Iwazaki* that is directed to an analyzer circuit as recited in the claims. As such, *Iwazaki* fails to anticipate the present invention.

Claim 7 is directed to a semiconductor integrated circuit having one or more functional circuit blocks and executing a set of instructions in a plurality of pipelines configured as a superscalar architecture, comprising a clock selector circuit that selects a different clock to be fed to each of the pipelines for each execution cycle. Based on the teachings set forth above, it is respectfully submitted that there is no teaching or suggest in

Iwazaki that is directed to these claim elements. Further, claim 7 additionally includes elements similar to those discussed above with regard to claim 6. As such, claim 7 is not anticipated by *Iwazaki* for the reasons set forth above with regard to claim 6, and for the reasons noted above.

Claim Rejections - 35 U.S.C. § 103(a) - *Iwazaki/Dowling*

With regard to the Examiner's rejection of claim 8, the Examiner admits that *Iwazaki* fails to teach or suggest a compiler that converts the instructions into a VLIW format and assigns a most suitable clock to each of the instructions in accordance with content thereof to be processed. The Examiner relies on the teachings of *Dowling* to cure the deficiencies of the teachings of *Iwazaki*, asserting *Dowling* teaches this compiler, citing to col. 9, line 51 - col. 10, line 66. Applicants respectfully disagree with the Examiner's characterization of this reference.

The disclosure of *Dowling* is directed to a split embedded DRAM processor. While *Dowling* discloses a VLIW extension processor comprising a set of at least one functional unit which receives one or more instructions for execution in a given clock cycle (col. 10, lines 10-13), these teachings are insufficient to teach or suggest the present invention as set forth in claim 8.

The invention set forth in claim 8 recites, *inter alia*, a semiconductor integrated circuit having one or more functional circuit blocks and executing a set of instructions in a plurality of processing sections configured as a VLIW architecture, comprising a compiler that converts the instructions into a VLIW format and assigns a most suitable clock to each of the instructions in accordance with content thereof to be processed and a clock selector circuit that selects the clock assigned to each instruction by the compiler so that the selected clock is fed to a corresponding processing section for each execution cycle, wherein the semiconductor integrated circuit is configured so as to provide each of the processing sections with an independent clock that enables the processing sections to operate at a frequency different from each other according to a load of the instructions to be executed simultaneously.

There is no teaching or suggestion in *Dowling* that is directed to at least the compiler of the present invention and, for the reasons set forth above, *Iwazaki*'s teachings are insufficient to teach the clock supply circuit and the clock selector circuit as set forth in the claim. As such, the references cited by the Examiner fail to render claim 8 obvious.

As claim 10 contains elements similar to those discussed above with regard to claim 8, claim 10 is not obvious for the reasons set forth above with regard to claim 8. Further, there is no teaching or suggestion in *Dowling* that is directed to a compiler that determines a most appropriate clock for each instruction according to content thereof to be executed and writes information thereof thus determined to prescribed bits of a compiled instruction code, as asserted by the Examiner. As none of the references cited by the Examiner, either alone or in combination, assuming these references are combinable, teach or suggest all of the claim elements, it is respectfully requested that the outstanding rejection be withdrawn.

Claim Rejections - 35 U.S.C. § 103(a) - *Iwazaki/Kapusta et al.*

In support of the Examiner's rejection of claim 9, the Examiner relies on his rejection of claim 6 to teach all of the elements except for the clock selector circuit having a hierarchically arranged clock selector architecture in which clock branches are arranged hierarchically in accordance with frequency of use of the clocks. The Examiner relies on *Kapusta et al.* to cure the deficiencies of the teachings of *Iwazaki*, citing to col. 5, lines 41-51 and col. 7, lines 5-18.

Applicants disagree that *Iwazaki* teaches the clock selector circuit of the present invention for the reasons set forth above with regard to claim 6. As *Kapusta et al.* fails to teach or suggest this claim element, the present invention is not rendered obvious based upon the teachings of the references as cited by the Examiner.

Conclusion

Should there be any outstanding matters that need to be resolved in the present application, the Examiner is respectfully requested to contact Catherine M. Voisinet (Reg. No. 52,327) at the telephone number of the undersigned below, to conduct an interview in an effort to expedite prosecution in connection with the present application.

If necessary, the Commissioner is hereby authorized in this, concurrent, and future replies, to charge payment or credit any overpayment to Deposit Account No. 02-2448 for any additional fees required under 37 C.F.R. §§ 1.16 or 1.17; particularly, extension of time fees.

Respectfully submitted,

BIRCH, STEWART, KOLASCH & BIRCH, LLP

By [illegible]
Terrell C. Birch, #19,382

P.O. Box 747
Falls Church, VA 22040-0747
(703) 205-8000

TCB/CMV/jdm
2936-0142P

Attachment: Appendix A - "Instruction Execution Cycle"

(Rev. 02/12/2004)

D1 - 1

Instruction Execution Cycle

Contents

Introduction

At the grass roots of all modern processors is a process which has basically remained unchanged since the inception of computers, that of the instruction execution cycle.

This document will endeavour to explain the process of the instruction execution cycle. Should the reader wish to refer to additional background data from which the majority of the information contained in this document has been sourced, please refer to the section titled Biographical Information.

To steamline the learning process, a number of very resourcefull web sites have been included in the Hot Links section of this document.

Components

All computers can be summarised with just two basic components: (a) primary storage or memory and (b) a central processing unit or CPU. The CPU is the "brains" of the computer. Its function is to execute programs which are stored in memory. This procedure is accomplished by the CPU fetching an instruction stored in memory, then executing the retreived (fetched) instruction within the CPU before proceeding to fetch the next instruction from memory. This process continues until they are told to stop.

The following illustration (Figure 1) summarises this continuous process of fetching and execution of instructions.

D1 - 2

Fetch / Execute Cycle

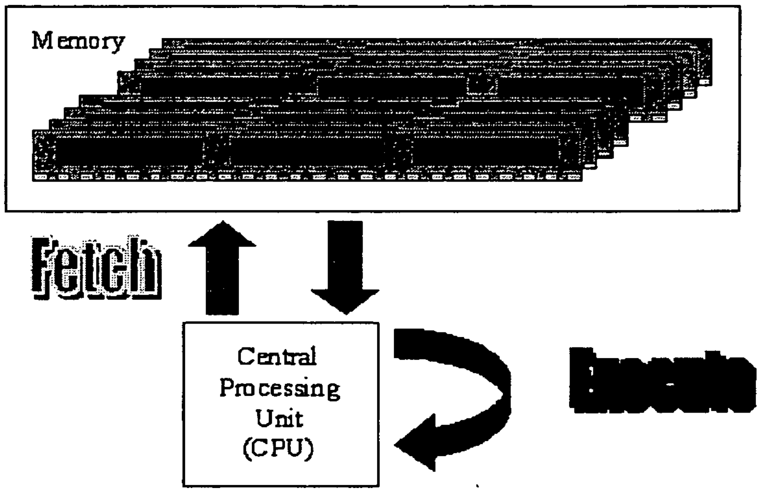


Figure 1: A simplified two step instruction cycle

What is the Instruction Execution Cycle?

All computers have an instruction execution cycle. A basic instruction execution cycle can be broken down into the following steps:

1. Fetch cycle
2. Execute cycle

Each of these two steps will be discussed in greater detail in the following sections.

Although we have been concentrating on the CPU and memory, there are additional components in a computer such as the I/O modules which can interact with the processor. In an improved instruction execution cycle, we can introduce a third cycle known as the interrupt cycle. Figure 2 illustrates how the interrupt cycle fits into the overall cycle.

D1-3

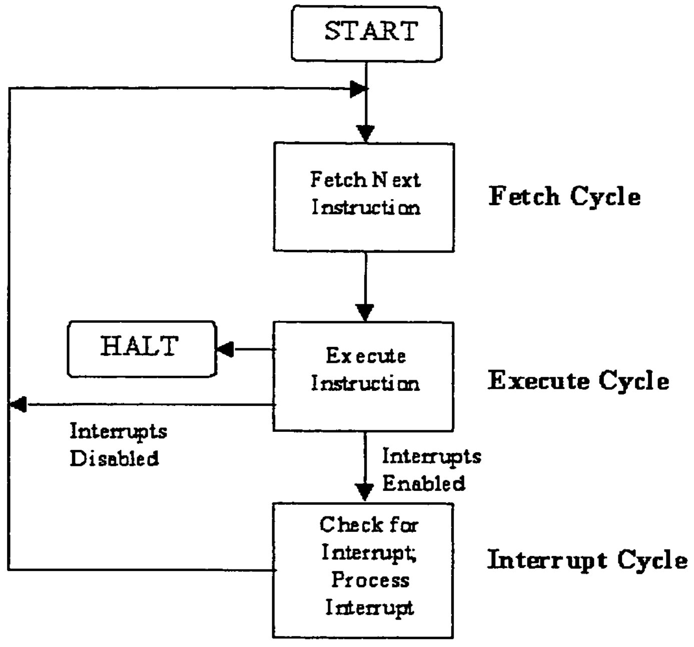


Figure 2: Instruction cycle with Interrupts (Stallings, 1995)

Back to top

Fetch Cycle

To start off the fetch cycle, the address which is stored in the program counter (PC) is transferred to the memory address register (MAR). The CPU then transfers the instruction located at the address stored in the MAR to the memory buffer register (MBR) via the data lines connecting the CPU to memory. This transfer from memory to CPU is coordinated by the control unit (CU). To finish the cycle, the newly fetched instruction is transferred to the instruction register (IR) and unless told otherwise, the CU increments the PC to point to the next address location in memory.

D1-4

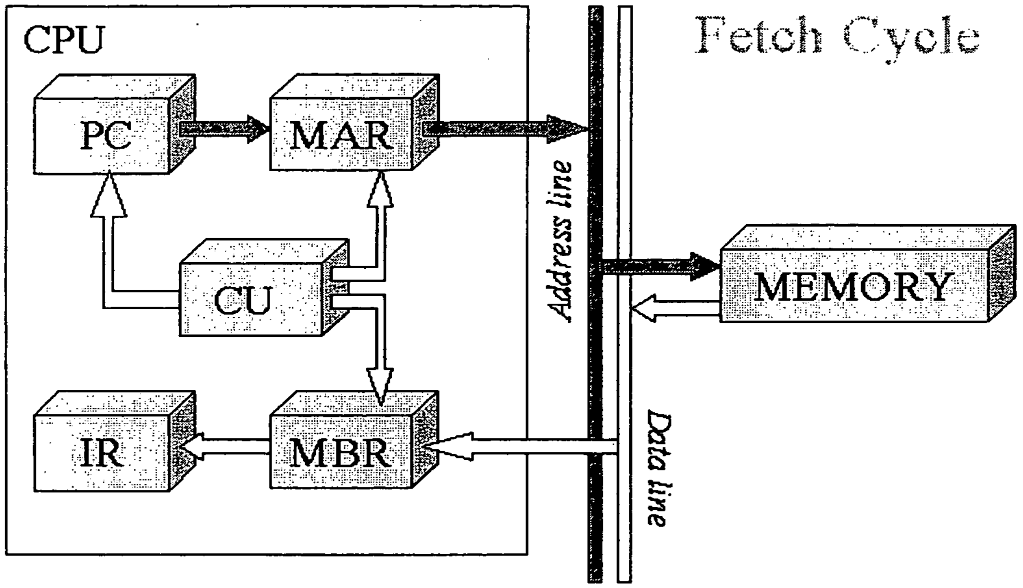


Figure 3: An animated fetch cycle

The illustrated fetch cycle above (Figure 3) can be summarised by the following points:

1. PC => MAR
2. MAR => memory => MBR
3. MBR => IR
4. PC incremented

After the CPU has finished fetching an instruction, the CU checks the contents of the IR and determines which type of execution is to be carried out next. This process is known as the decoding phase. The instruction is now ready for the execution cycle.

Back to top

Execute Cycle

Once an instruction has been loaded into the instruction register (IR), and the control unit (CU) has examined and decoded the fetched instruction and determined the required course of action to take, the execution cycle can commence. Unlike the fetch cycle and the interrupt cycle, both of which have a set instruction sequence, the execute cycle can comprise some complex operations (commonly called opcodes).

The actions within the execution cycle can be categorised into the following four groups:

1. *CPU - Memory*: Data may be transferred from memory to the CPU or from the CPU to memory.
2. *CPU - I/O*: Data may be transferred from an I/O module to the CPU or from the CPU to an I/O module.
3. *Data Processing*: The CPU may perform some arithmetic or logic operation on data via the arithmetic-logic unit (ALU).

4. *Control*: An instruction may specify that the sequence of operation may be altered. For example, the program counter (PC) may be updated with a new memory address to reflect that the next instruction fetched, should be read from this new location.

For simplicity reasons, the following examples (illustrated in Figure's 4 and 5) will deal with two operations that can occur. The [LOAD ACC, memory] and [ADD ACC, memory], both of which could be classified as memory reference instructions. Instructions which can be executed without leaving the CPU are referred to as non-memory reference instructions.

LOAD ACC, memory

This operation loads the accumulator (ACC) with data that is stored in the memory location specified in the instruction. The operation starts off by transferring the address portion of the instruction from the IR to the memory address register (MAR). The CPU then transfers the instruction located at the address stored in the MAR to the memory buffer register (MBR) via the data lines connecting the CPU to memory. This transfer from memory to CPU is coordinated by the CU. To finish the cycle, the newly fetched data is transferred to the ACC.

The illustrated LOAD operation (Figure 4) can be summarised in the following points:

1. IR [address portion] => MAR
2. MAR => memory => MBR
3. MBR => ACC

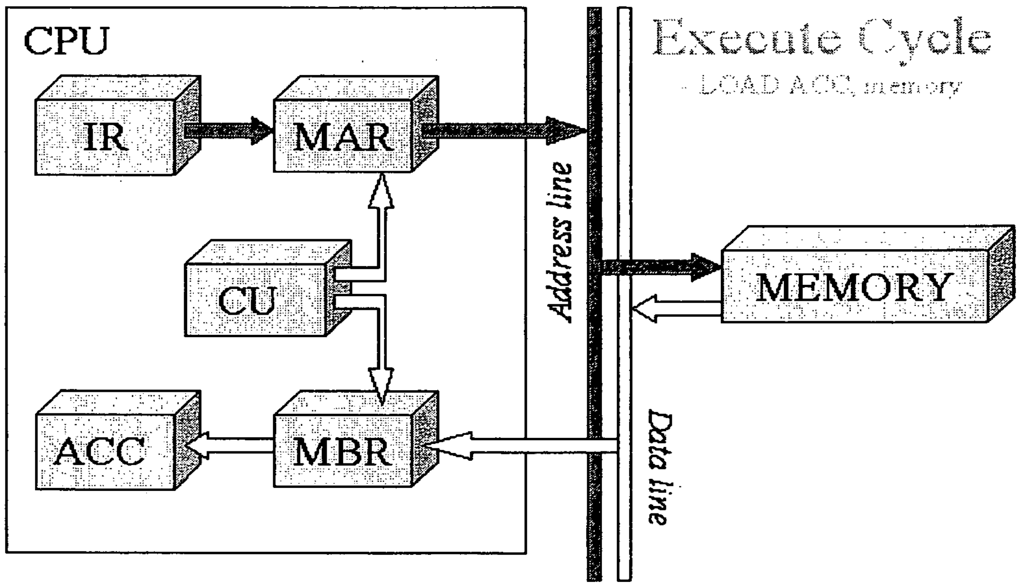


Figure 4: Animated Execute Cycle [LOAD ACC, memory] operation

ADD ACC, memory

This operation adds the data stored in the ACC with data that is stored in the memory location specified in the instruction using the ALU. The operation starts off by transferring the address portion of the instruction from the IR to the MAR. The CPU then transfers the instruction located at the address stored in the MAR to

D1-6

the MBR via the data lines connecting the CPU to memory. This transfer from memory to CPU is coordinated by the CU. Next, the ALU adds the data stored in the ACC and the MBR. To finish the cycle, the result of the addition operation is stored in the ACC for future use.

The illustrated ADD operation (Figure 5) can be summarised in the following points:

1. IR [address portion] => MAR
2. MAR => memory => MBR
3. MBR + ACC => ALU
4. ALU => ACC

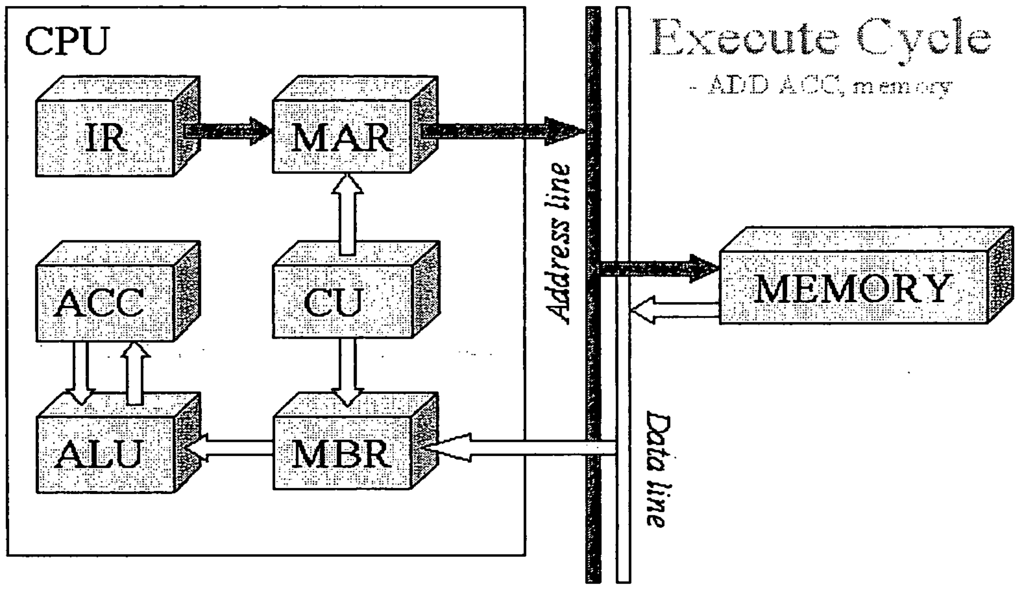


Figure 5: Animated Execute Cycle [ADD ACC, memory] operation

After the execution cycle completes, if an interrupt is not dedected, the next instruction is fetched and the process starts all over again.

Back to top

Interrupt Cycle

An interrupt can be described as a mechanism in which an I/O module etc., can break the normal sequential control of the central processing unit (CPU). Table 1 below, summarises the most common form of interrupts that the CPU can receive.

The main advantage of using interrupts is that the processor can be engaged in executing other instructions while the I/O modules connected to the computer are engaged in other operations.

Program	Generated by some condition that occurs as a results of an instruction execution, such as arithmetic overflow,

D1 - 7

	division by zero, attempt to execute am illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure such as power failure or memory parity error.

Table 1: Classes of Interrupts (Stallings, 1995)

Up until now we have dealt with the instruction execution cycle on the hardware level. When interrupts are introduced, the CPU and the operating system driving the system, is responsible for the suspension of the program currently being run, as well as restoring that program at the same point before the interrupt was detected. To handle this, an interrupt handler routine is executed. This interrupt handler is usually built into the operating system.

Before the interrupt handler routine can ran, several processes must occur first. A typical sequence of events is illustrated in Figure 6 below. After the completion of the interrupt handler routine, the normal sequencial fetch / execute cycle begins.

D1-8

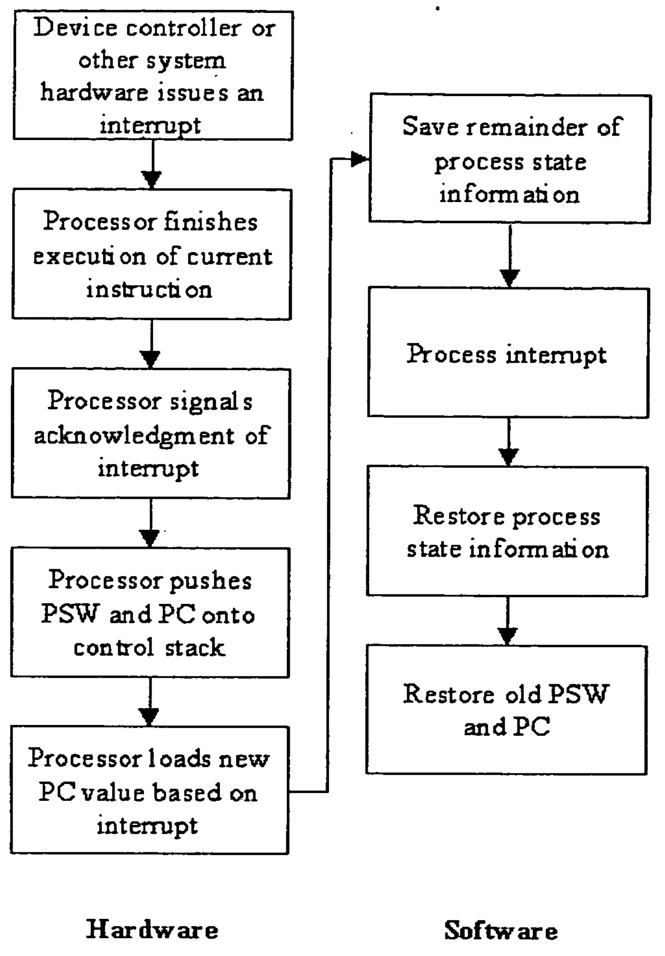

Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC
Hardware
Software

Figure 6: Sample interrupt processing (Stallings, 1995)

Back to top

Modern Processors

When central processing units (CPU's) were first developed they processed the first instruction before starting the second. For example, the processor fetched the first instruction, decoded it, then executed the fetched instruction, before fetching the second instruction and starting the process over again (see Figure 7). In a processor, such as the one just described, the CPU itself is the weak link. The external bus operates for at least one cycle (clock pulse) in the three, but has to wait the remaining cycles for the CPU.

Note: Figures 7, 8, 9 and 10 have been adapted from Tredennick, 1996.

D1-9

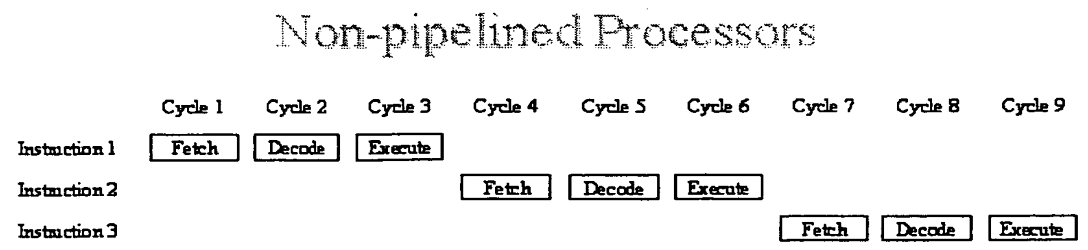


Figure 7: Instruction execution cycle in a non-pipelined processor

Modern processors on the other hand, have developed what are called pipelines. Pipelines are the most common implementation technique in a CPU today that increases the performance of the system. The idea behind the pipeline is that while the first instruction is being executed, the second instruction can be fetched, or in simple terms, instruction overlap.

The first pipelines to be introduced where a simple three-stage pipeline (see Figure 8). While this utilises all the resources of the system, conflict of resources can occur, resulting in instructions being held untill the previous instruction has completed its current stage. Apart from these minor hiccups, it is possible for the CPU to complete an instruction every cycle as opposed to the earlier processors which required three cycles per instruction.

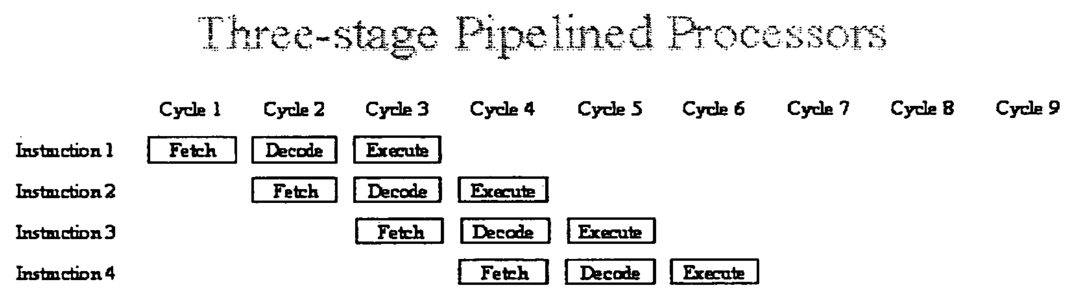
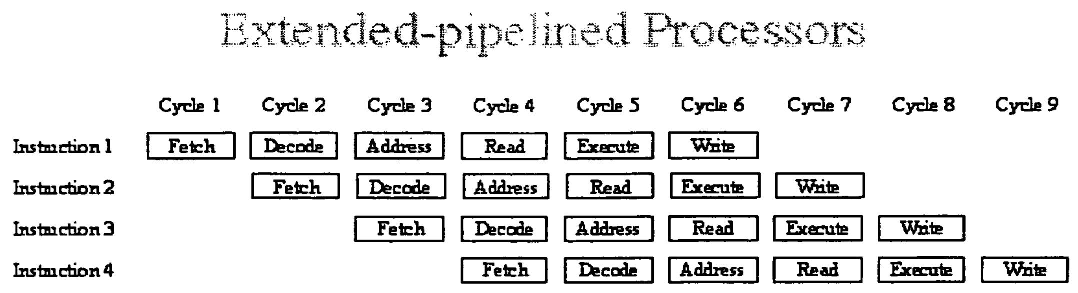


Figure 8: Instruction execution cycle in a three-stage pipelined processor

Figure 9 illustrates what is known as an extended pipeline. To overcome the delays associated with the three stage pipeline, modern processors have broken down the execute cycle into a number of phases, some have even broken down the fetch cycle in the fight to overcome delays in their processors. No matter how many phases the cycle is broken down to, the end result is that only one instruction can be completed every cycle.

D1 - 10

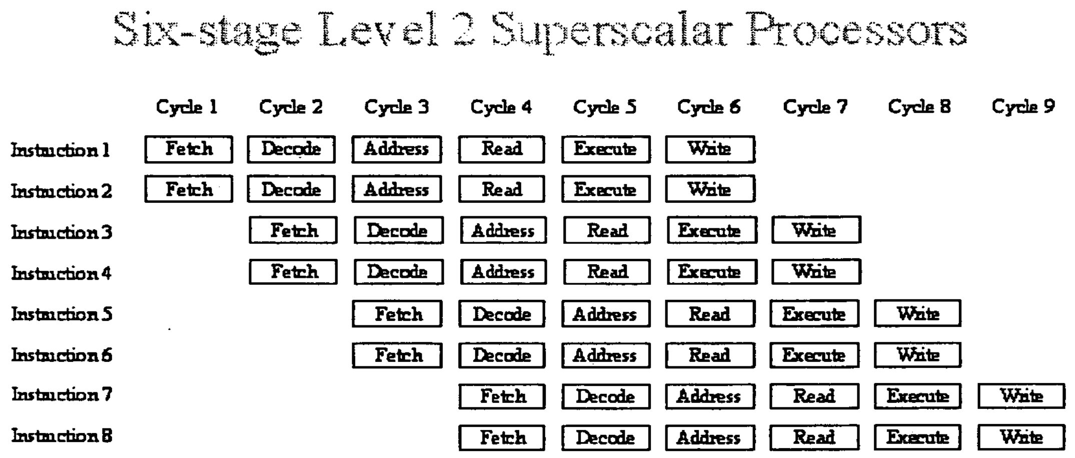


Extended-pipelined Processors

	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8	Cycle 9
Instruction 1	Fetch	Decode	Address	Read	Execute	Write			
Instruction 2		Fetch	Decode	Address	Read	Execute	Write		
Instruction 3			Fetch	Decode	Address	Read	Execute	Write	
Instruction 4				Fetch	Decode	Address	Read	Execute	Write

Figure 9: Instruction execution cycle in an extended-pipelined processor

Enter the world of superscalar pipeline, where more than one instruction can be issued per clock cycle. Intel describes their processors by different levels. For example a level 2 (L2) processor (Pentium) can issue two instructions per clock cycle and their level 3 (L3) processor (Pentium Pro) can issue 3. Figure 10 illustrates a level 2 superscalar pipelined processor. Other processors which are available today include Motorola's 68060 (L2) and DEC's Alpha 21164 chip (L4) to name a few.



Six-stage Level 2 Superscalar Processors

	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8	Cycle 9
Instruction 1	Fetch	Decode	Address	Read	Execute	Write			
Instruction 2	Fetch	Decode	Address	Read	Execute	Write			
Instruction 3		Fetch	Decode	Address	Read	Execute	Write		
Instruction 4		Fetch	Decode	Address	Read	Execute	Write		
Instruction 5			Fetch	Decode	Address	Read	Execute	Write	
Instruction 6			Fetch	Decode	Address	Read	Execute	Write	
Instruction 7				Fetch	Decode	Address	Read	Execute	Write
Instruction 8				Fetch	Decode	Address	Read	Execute	Write

Figure 10: Instruction execution cycle in a six-stage level 2 superscaler pipelined processor

Back to top

Glossary of Terms

Accumulator (ACC)

A register located on the central processing unit. The contents can be used by the arithmetic-logic unit for arithmetic and logic operations, and by the memory buffer register. Usually, all results generated by the arithmetic-logic unit end up in the accumulator.

Arithmetic-Logic Unit (ALU)

Performs arithmetic operations such as addition and subtraction as well as logical operations such as